

The Great IDS Debate : Signature Analysis Versus Protocol Analysis

Matthew Tanase 2003-02-05

The Great IDS Debate : Signature Analysis Versus Protocol Analysis

by [Matt Tanase](#)

last updated Feb. 5, 2003

Intrusion detection systems (IDS) have rapidly become a crucial component of any network defense strategy. Over the past few years, their popularity has soared as vendors have refined their results and increased performance capabilities. At the heart of intrusion detection systems lies the analysis engine. It reviews each packet, determines if it is malicious, and logs an alert if necessary – the core tasks of an IDS. Two different IDS techniques, each favored by separate and loyal camps, have emerged as the preferred engine behind the software. Despite the copious marketing material and [fiery online debates](#), each method has distinct strengths and weaknesses. In this article, we'll examine and compare the two different techniques: signature analysis and protocol analysis.

Traditional Signature Analysis and Protocol Analysis

Before exploring the signature and protocol analysis methodologies, let's briefly review the fundamental IDS concepts. These devices, similar to firewalls, inspect incoming and outgoing network traffic. Unlike firewalls, however, they do not alter the traffic flow by dropping or passing certain packets. Rather, they look for malicious traffic that may be indicative of an attack or other misuse and log an alarm with specific data for administrative review. This discussion will focus on the techniques an IDS employs to catch malicious traffic.

Signature analysis was the first method implemented in intrusion detection. It is based on the simple concept of string matching, also known as pattern matching. In string matching, an incoming packet is compared, byte by byte, with a single signature, a string of code that indicates a particular characteristic of malicious traffic. That signature might contain a key phrase or command that is often associated with an attack. If a match is found, an alert is generated. If not, the data in the packet is then compared to the next signature on the list. Once all of the signatures have been checked, the next packet is read into memory and the process begins again. Critics of signature analysis often refer to it, somewhat accurately, as "packet grepping", a derogatory reference to the Unix string matching tool `grep`. They consider this type of IDS engine to be extremely elementary; something could be duplicated using a simple Unix command.

The second method of analysis focuses on reviewing the strictly formatted data of network traffic, otherwise known as protocols. Each packet is wrapped in predefined layers of different protocols. IDS authors, recognizing this, implemented engines that unwrap and inspect these layers, according to the protocol standards or RFC. Each wrapper has several fields with expected or normal values. Anything that violates or is outside of these standards is likely malicious. The IDS inspects each field of the different protocols of an incoming packet: IP, TCP, and UDP. If something violates a protocol rule, for instance, if it contains an unexpected value, an alert is generated. Protocol analysis uses a detailed knowledge of expected or normal packet field values to discover malicious traffic. This differs greatly from signature analysis, which uses known traits of an attack, to generate an alert.

Modern Day Signature and Protocol Analysis

The core concepts above describe rudimentary implementations of the signature and protocol analysis models. However, each has evolved significantly. Most products that employ signature analysis also use basic protocol analysis. Layers 3 (network) and 4 (transport) of the OSI model, which contain IP, TCP and UDP, are all examined. Current IDS implementations understand these protocols and the expected values of their respective fields. The signatures in use today often examine multiple fields from different protocols, such as source address, destination port, or TCP flags. Additionally, these systems, based on the decoded protocol results, know where to look for the packet payload, which is searched for specific strings that may indicate an attack. Such strings might include the code of an exploit or a command that would indicate a malicious attack. For example, the following [Snort](#) rule is for a Telnet login failure:

```
alert tcp $HOME_NET 23 -> $EXTERNAL_NET any (msg:"TELNET Bad Login"; content: "Login failed"; nocase; flow:from_server,established; classtype:bad-unknown; sid:492; rev:5;)
```

It's important to note that the engine actually searches the payload for the "Login failed" string, as is evidenced by the "content:" field. If this string is found, or matched, an alert is generated.

IDSs based on protocol analysis have evolved as well. In addition to the layer 3 and 4 decoding discussed above, they also focus on a detailed analysis of layer 7 (application) protocols. Programmers have implemented several of the most popular protocols, such as HTTP, SMTP, and Telnet, all of which can be examined in detail for violations and abnormalities. These

applications can now anticipate the expected response during key parts of a given transaction. Therefore, anomalies such as unexpected values, unusually large or small packets, or strange options, all of which may signal an attack, will be detected.

Strengths and Weaknesses: Signature Analysis

Signature analysis systems have a few key strengths. They are very fast, since packet matching is a relatively non-processor intensive task. The rules are easy to write and understand, as well as very customizable. Additionally, there is fantastic community support for rapidly generating signatures for new alerts and warnings. These systems excel at catching low level, simple attacks since they tend to employ prepackaged exploits that are easy to recognize. Lastly, signature-based analysis conveys exactly what has happened very well, since it takes a very specific event to trigger an alert.

On the flip side, signature-based IDS has definite weaknesses. While initially very fast, this performance edge slips away as the ruleset grows. This is particularly problematic as the ruleset can grow very fast – basically, for each attack or exploit that is created by attackers, a new rule must be created to detect it. Despite data normalizers and packet reassembly, both of which eliminate some evasion techniques, uncountable variations of attacks can slip by a signature-based system. Application level attacks such as Unicode, multiple variations similar to those found in SNMP community strings, and evasion programs that morph shell code like ADMutate can cause serious problems for any signature system. The slightest variation in an attack is often enough to defeat a signature. The only solution is more rules, which eats away at performance and increases complexity. And as any IDS admin can confirm, such systems generate loads of false positives due to the sometimes simplistic nature of their packet evaluations and the sheer number of rules in place. This leads to an associated problem: because they work by comparing a list of signatures to the packet data, signature-based attacks can only catch attacks that are known and for which signatures have been created.

Strengths and Weaknesses: Protocol Analysis

The case for protocol analysis is very similar: it has distinct positives and negatives, most of which are in direct opposition to signature analysis. Due to the preprocessors required for advanced protocol examination, protocol analysis can be fairly slow to begin with. Furthermore, the rules for a protocol system are difficult to write and understand. In some ways, it's extremely vendor dependent, since a processor is required for each protocol that is to be

reviewed, and the rules are relatively complex. The increasingly complex and frequently ignored protocol standards and RFCs pose an additional problem for IDS developers, making it difficult to write accurate processors.

While at first glance protocol-based IDSs are slower than signature-based systems, they more than make up ground in terms of scalability and performance as signature-based rulesets grow. Furthermore, since they search for generic violations, protocol analysis engines can often catch zero-day exploits, something that is impossible for a signature system; unfortunately, they can sometimes miss obviously deviant events, such as a root Telnet session, that do not violate any protocol. Protocol-based systems keep the false alarms to a minimum, since they log real violations. Unfortunately, they often don't provide enough information. Instead, they merely note the anomaly, placing the burden of investigation on the admin.

The IDS Environment

Of course, the effectiveness of an IDS depends upon the environment in which it will be employed. Monitoring a large, diverse network is very different from smaller, homogenous environments. Signature analysis models are best suited for average-sized networks looking to catch standard threats. Administrators can draw on the fantastic community support for releasing updated signatures, and performance is not a crucial factor. However, a bigger, ever-changing network would likely benefit from some of the strengths of a protocol analysis system: performance, minimal false positives, and generalized alerts. A better, but more expensive solution, would be having an IDS deployment containing two machines, each employing a different model. Unfortunately, very few can afford such a luxury.

The Future: A Complementary Model?

Without a doubt, anyone choosing an IDS based on one of these techniques has several factors to consider. Each model excels in different arenas. Fortunately, it appears as though we're headed in the direction of a reconciliation between the two divergent methods. The engineers and programmers behind these systems recognize the obvious strengths and weaknesses of each approach. As can be expected, the developers are attempting to pull together the best components of each approach in order to provide a more robust product – a fact that is evident in several of the more recent IDS offerings. Currently, almost all of the protocol-based offerings perform pattern matching at some point in the application level decode. There are IDS systems that, even though they perform protocol analysis, also allow the user/operator the ability to

create signatures for particular traffic. We can expect to see more of this as well. Similarly, signature-based systems are bundling application processors to more effectively recognize attacks.

Ultimately, we're moving toward a much more thorough design, one that will statefully examine the transactions and know what sort of client and server response to expect at a given point in the process. Hopefully, this competition between the camps of the two techniques will eventually reap dividends for the end user, and soon.

Conclusion

While the two intrusion detection camps, signature and protocol analysis, seem, at first, vastly different, a more philosophical study of the problem at hand reveals encouraging similarities. In the end, these security tools examine formatted data for attacks and anomalies. While the two techniques initially seemed unrelated, it is obvious by the overlap in modern product offerings that they can be very similar. Exploiting the inherent strengths of each approach while avoiding the weaknesses will lead to one end – a superior product.

[Matt Tanase](#) is President of [Qaddisin](#). He and his company provide nationwide security consulting services. Additionally, he produces [The Security Blog](#), a daily weblog dedicated to network security.

Relevant Links

[FOCUS-IDS Discussion](#) – several insightful comments from IDS authors and experts are included in this thread.

[SecurityFocus](#)

[One of These Things is Not Like the Other: the State of Anomaly Detection](#)

[Matt Tanase](#), [SecurityFocus](#)

[The Future of Intrusion Detection Systems](#)

[Matt Tanase](#), [SecurityFocus](#)

[Network Intrusion Detection Signatures - A Five-Part Series](#)

[Karen Kent](#), [SecurityFocus](#)

[Privacy Statement](#)

Copyright 2006, SecurityFocus